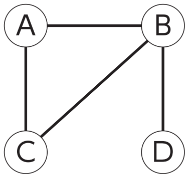
上級編
第 45 講 グラフ理論

例題

グラフ理論とは，物事の間の関係を数学的に表現し，分析するための理論である。
このグラフ理論を用いた簡単な例として，友人関係を調べるプログラムを以下の条件で作成する。

・A，B，C，D の 4 人の関係を表したグラフは次のようになる。



・線でつながっている人が友人関係となる。
・例えば，A は B と C と友達であるが，D とは友達ではない。
・友人関係ではない場合を 0，友人関係である場合を 1 とする表を作成すると次のようになる。

	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	0
D	0	1	0	0

・これを 2 次元配列（配列名：Data）とした場合，2 行 3 列目のデータを取得する場合は添字は 0 から始まるので，「Data[1][2]」というように記述する。
・name_index ＝ Human.index(name) という命令は，配列 Human の中で変数 name の値と一致する要素の添字（インデックス）を取得し，変数 name_index に代入するものである。（例：Human=['A','B','C','D']，name='A' のとき，Human.index(name) は 0）

以下のプログラムの空欄に入る最も適切なものを，あとの①～④のうちから一つ選びなさい。

```
Data = [
#  A  B  C  D
  [0, 1, 1, 0], # A
  [1, 0, 1, 1], # B
  [1, 1, 0, 0], # C
  [0, 1, 0, 0]  # D
]
Human = ['A', 'B', 'C', 'D']
Friends = []
name = input('AからDの中で，友人関係を知りたい人を入力してください。(例:B): ')
name_index = Human.index(name) # 入力された人に対応する添字を取得

for i in range(len(Data)):
  if [　　　　　　]:
    Friends.append(Human[i])

# 結果表示（カンマ区切りで表示）
print(name, 'の友達は', Friends)
```

① `Data[name_index][i] == 0`

② `Data[name_index][i] == 1`

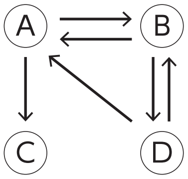
③ `Data[name_index][i] != 1`

④ `Data[name_index] == 0`

上級編　第 45 講　確認問題

グラフ理論の「有向グラフ」とは，関係を示す線に矢印がついている関係図のことで，物事の関係性を把握するのに適している。
この有向グラフの考え方を用いて，SNS のフォロー関係を調べるプログラムを以下の条件で作成する。

- A，B，C，D の４人のフォロー関係を表したグラフは次のようになる。



- グラフ内で，矢印が指し示している先がフォローしている人となる。
- 例えば, D は A と B をフォローしていて, B からフォローされている。
- フォローしていない場合を 0，フォローしている場合を 1 とする表を作成すると次のようになる。

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	0	0	0	0
D	1	1	0	0

- これを２次元配列（配列名：Data）とした場合，２行３列目のデータを取得する場合は添字は０から始まるので，「Data[1][2]」というように記述する。
- name_index = Human.index(name) という命令は，配列 Human の中で変数 name の値と一致する要素の添字を取得し，変数 name_index に代入するものである。

以下のプログラムの空欄に入る最も適切なものを，あとの①～④のうちから一つ選びなさい。

```
Data = [
  [0, 1, 1, 0], # A
  [1, 0, 0, 1], # B
  [0, 0, 0, 0], # C
  [1, 1, 0, 0]  # D
]

Human = ['A', 'B', 'C', 'D']
ToFollow = [] # フォローしている人を格納
FromFollow = [] # フォローされている人（フォロワー）を格納

name = input('A から D の中で，フォロー関係を知りたい人を入力してください。（例 :B）: ')
name_index = Human.index(name) # 入力された人に対応する添字を取得

for i in range(len(Data)):
  if Data[name_index][i] == 1:
    ToFollow.append(Human[i])
  if ____________:
    FromFollow.append(Human[i])

# 結果表示
print(name, ' がフォローしている人は ')
print(ToFollow)
print(name, ' をフォローしている人は ')
print(FromFollow)
```

① `Data[name_index][i] == 0`

② `Data[i][name_index] != 1`

③ `Data[i][name_index] == 1`

④ `FromFollow[i] == name`

上級編　例題　解答

第34講　コインの枚数

例題　解答：④

第35講　駐車料金の判定

例題　解答：④

第36講　割引のある料金

例題　解答：④

第37講　テストの集計

例題　解答：①

第38講　じゃんけん

例題　解答：①

第39講　サイコロの出目

例題　解答：③

第40講　中央値の判定

例題　解答：①

第41講　素数判定

例題　解答：③

第42講　素因数分解

例題　解答：②

第43講　倍数の判定・FizzBuzz

例題　解答：④

第44講　バブルソート・選択ソート

例題　解答：③

第45講　グラフ理論

例題　解答：②

第46講　待ち行列

例題　解答：②

第47講　パリティチェック

例題　解答：①

第48講　すごろくゲーム

例題　解答：④

上級編　第 45 講　確認問題　解答

正解：③

解説：

このプログラムでは，⑴入力した人がフォローしている人，⑵入力した人をフォローしている人の２パターンを求めて表示する必要がある。

まず，16 行目の if 文は，⑴を求める（入力した人は誰をフォローしているか）ための判定式である。

そのため，次の if 文である 18 行目では，⑵を求める（入力した人を誰がフォローしているか）ための条件式を指定する必要がある。

よって，２次元配列の１つ目の添字には，表の上から何行目であるかを指定する変数 i が，２つ目の添字には，入力した人の列を指定する [name_index] が当てはまる。

よって，正解は ③ `Data[i][name_index] == 1` である。

```
Data = [
  [0, 1, 1, 0], # A
  [1, 0, 0, 1], # B
  [0, 0, 0, 0], # C
  [1, 1, 0, 0]  # D
]

Human = ['A', 'B', 'C', 'D']
ToFollow = [] # フォローしている人を格納
FromFollow = [] # フォローされている人（フォロワー）を格納

name = input('A から D の中で，フォロー関係を知りたい人を入力してください。（例 :B）： ')
name_index = Human.index(name) # 入力された人に対応する添字を取得

for i in range(len(Data)):
  if Data[name_index][i] == 1:
    ToFollow.append(Human[i])
  if Data[i][name_index] == 1:
    FromFollow.append(Human[i])

# 結果表示
print(name, ' がフォローしている人は ')
print(ToFollow)
print(name, ' をフォローしている人は ')
print(FromFollow)
```

実行結果

```
A から D の中で、フォロー関係を知りたい人を入力してください。
（例 :B）: B
B がフォローしている人は
['A', 'D']
B をフォローしている人は
['A', 'D']
```